

Учебный модуль
Программирование. Начальный курс

2012 - 2013

СОДЕРЖАНИЕ

1. Знакомство со средой программирования	3
1.1. Среда программирования PascalABC	3
1.2. О языке Паскаль	3
1.3. Первая программа	3
1.4. Структура программы и правила ее записи.....	7
1.5. Переменные, простые типы данных, оператор присваивания	9
1.6. Консольный ввод-вывод	11
2. Ветвление	15
2.1. Условный оператор.....	15
2.2. Оператор вариантов	16
3. Организация повторения.....	18
3.1. Детерминированные циклы.....	18
3.2. Итеративные циклы с предусловием	20
3.3. Итеративные циклы с постусловием	23
4. Основные операции работы с текстовыми файлами	26
5. Задачи для самостоятельного решения	29
Литература.....	31

1. Знакомство со средой программирования

1.1. Среда программирования PascalABC

Система PascalABC основана на языке Delphi (Object Pascal) и предназначена для обучения программированию на языке Паскаль; ориентирована на школьников и студентов младших курсов.

Система разработана на факультете математики, механики и компьютерных наук Южного федерального университета. Автор системы PascalABC – доцент механико-математического факультета Ростовского государственного университета (ЮФУ) Станислав Станиславович Михалкович. Многие концепции в PascalABC сознательно упрощены, скорость работы программ невысока, но для начала изучения программирования эта среда подходит хорошо.

PascalABC.NET работает на платформе Microsoft .NET и содержит все основные элементы современных языков программирования¹.

1.2. О языке Паскаль

Первая версия языка программирования Pascal была разработана на кафедре информатики Стэнфордского университета швейцарским ученым Никлаусом Виртом в 1968 г. (в некоторых источниках 1970 г.). Паскаль – язык со строгой типизацией и интуитивно понятным синтаксисом.

Существуют разные среды для разработки программ на языке Паскаль: Turbo Pascal, Borland Pascal (80-е гг., ориентирована на MS DOS, устарела), Delphi (90-е гг., добавлены объектно-ориентированные расширения – язык Object Pascal, Delphi; для быстрого создания приложений под Windows, богатые возможности, но сложна для начинающих), Free Pascal (многоплатформенный компилятор с открытым исходным кодом), TMT Pascal и т. д. Мы будем работать в системе PascalABC.NET.

1.3. Первая программа

Задача 1. Написать программу решения произвольного квадратного уравнения $ax^2+bx+c=0$.

Алгоритм:

- 1) Организовать ввод данных – коэффициентов a , b , c .
- 2) Вычислить дискриминант D по формуле $D = b^2 - 4 \cdot a \cdot c$.

¹ Скачать PascalABC.Net можно по ссылке: <http://pascalabc.net/ssyilki-dlya-skachivaniya.html>.

- 3) Если $D < 0$, то уравнения не имеет решений.
- 4) Если $D = 0$, то уравнение имеет 2 равных решения. Вычислить их по формуле

$$x = -\frac{b}{2a}.$$

- 5) Если $D > 0$, то уравнение имеет 2 различных решения. Вычислить их по формулам:

$$x_1 = \frac{-b + \sqrt{D}}{2a}, \quad x_2 = \frac{-b - \sqrt{D}}{2a}.$$

- 6) Вывести результаты на экран.

Замечание: функция `sqrt(D)` находит корень квадратный из переменной `D`, функция `sqr(D)` возводит `D` в квадрат.

Для пояснения программного кода рекомендуется использовать *комментарии*. Они никак не влияют на выполнение программы. Комментарии в Паскале могут быть однострочные (обозначаются двойным слешем «`/`») и многострочные (фигурные скобки «`{ }`»).

Программа.

```

Program kwur; {задаём любое название программе и после
              каждой команды обязательно точка с
              запятой}
Var a, b, c, D, x, x1, x2 : Real;
              {описание всех переменных и их типов,
              которые будут использованы в программе}
BEGIN        {начало программы}
  Writeln('Программа для решения уравнений вида
ax^2+bx+c=0'); { вывод текста на экран}

  Writeln;      { перевод строки }
  Write('Введите a '); { вывод текста на экран }
  Readln(a);    { ввод данных с клавиатуры }
  Write('Введите b '); { вывод текста на экран }
  Readln(b);    { ввод данных с клавиатуры }
  Write('Введите c '); { вывод текста на экран }
  Readln(c);    { вывод текста на экран }

```

```

Writeln;                { перевод строки }
D:=b*b-4*a*c;           {вычислить значение дискриминанта D }
Writeln('Дискриминант = ',D);
                        {вывод текста 'Дискриминант = ' и
                        значения переменной D на экран }

If D=0 Then Begin
    Writeln('D=0, уравнение имеет 2 равных корня');
    x:=-b/(2*a);
    Writeln('x = ',x);
End;                    {если D=0, то найти x:=-b/(2*a) и
                        вывести результат на экран }

If D<0 Then Writeln('D<0, нет решений');
                        {если D<0, то вывести информацию об
                        отсутствии решений }

If D>0 Then Begin      { если D>0 то... }
    Writeln('D>0, уравнение имеет 2 разных корня');
    x1:=( -b+sqrt(d) ) / (2*a);
                                {Вычислить значение X1 }
    Writeln('X1=',x1);        { вывести его на экран }
    Writeln;                  { перевод строки }
    x2:=( -b-sqrt(d) ) / (2*a);
                                { Вычислить значение X2 }
    Writeln('X2=',x2);        { вывести его на экран }
    End;
    Writeln;                  { перевод строки }
End.                        {Конец программы }

```

Запуск и остановка программы

Для запуска программы в текущем окне редактора следует нажать клавишу **F9**.

Программа вначале компилируется во внутреннее представление, после чего, если не найдены ошибки, программа начинает выполняться. При выполнении программы кнопка запуска программы становится неактивной, кнопка останова программы, наоборот, активной и в строке статуса отображается информация «Программа выполняется».

Выполнение программы можно в любой момент прервать нажатием комбинации клавиш **Ctrl-F2**. При этом в окне вывода появится сообщение «Программа прервана пользователем».

Под окном редактора расположено окно вывода. Оно предназначено для вывода данных процедурами `Write` и `Writeln`, а также для вывода сообщений об ошибках и предупреждений во время работы программы.

Окно вывода может быть скрыто. Клавиша **F5** показывает / скрывает окно вывода. Для скрытия окна вывода используется также клавиша **Esc**.

Окно вывода обязательно открывается при любом выводе в него.

Для очистки окна вывода следует нажать комбинацию клавиш **Ctrl-Del** или кнопку.

Окно вывода		
Программа для решения уравнений вида $ax^2+bx+c=0$ Введите a 1 Введите b 1 Введите c 2 Дискриминант = -7 D<0 нет решений	Программа для решения уравнений вида $ax^2+bx+c=0$ Введите a 1 Введите b 2 Введите c 1 Дискриминант = 0 D=0 уравнение имеет 2 равных корня x = -1	Программа для решения уравнений вида $ax^2+bx+c=0$ Введите a 1 Введите b 3 Введите c 1 Дискриминант = 5 D>0 уравнение имеет 2 разных корня X1=-0.381966011250105 X2=-2.6180339887499

Для отладки программ предназначен режим пошагового выполнения. Для выполнения одного шага (одной строки) программы следует нажать клавишу **F8**.

Прервать программу, находящуюся в режиме пошагового выполнения, можно с помощью комбинации клавиш **Ctrl-F2**. Если программа находится в режиме пошагового выполнения, то ее можно выполнить до конца, нажав **F9**.

1.4. Структура программы и правила ее записи

Программа имеет следующий вид:

```
Program <имя программы>;  
<раздел подключения модулей>  
<раздел описаний>  
Begin  
    <операторы>  
End.
```

Правила записи программ

- 1) Текст программы записывается при помощи *латинских букв, цифр и знаков*. Допускаются *прописные и строчные буквы*.
- 2) Программа начинается с заголовка – слова Program и следующего за ним названия программы. Это строка не является обязательной.
- 3) Раздел подключения модулей начинается со служебного слова Uses, за которым следует список имен модулей, перечисляемых через запятую.
- 4) Раздел описаний может включать разделы описания переменных, констант, типов, процедур и функций, которые следуют друг за другом в произвольном порядке.
- 5) Между служебными словами (операторными скобками) Begin и End записывается алгоритм решения задачи.
- 6) Раздел подключения модулей и раздел описаний могут отсутствовать.
- 7) Операторы разделяются точкой с запятой.
- 8) Для удобочитаемости каждый оператор располагается в отдельной строке, хотя это не обязательно.
- 9) Перечисляемые объекты разделяются запятой.

Для обозначения программы и ее объектов применяются **имена**, которые могут состоять из **любого количества букв или цифр**, но должно начинаться с буквы. В имя можно включать **знак подчеркивания** (например, Prim_1).

В качестве имен нельзя использовать зарезервированные слова, которые используются для обозначения операторов и других элементов языка программирования и во всех программах имеют одинаковый смысл.

Задача 2. Написать программу, которая складывает два числа и выводит сумму на экран.

Примечание. Обратите внимание на оформление текста программы.

```

Program Sum_2;    {Задача. Вычислить сумму двух чисел и
                  вывести ее на экран.
                  Решение. Иванов Петр, 10 А класс.}

Var
  number1, number2, result : integer;
Begin            {признак начала программы}
  number1 := 3; {присваиваем переменной number1 значение 3}
  number2 := 4; {присваиваем переменной number2 значение 4}

  {складываем значения переменных number1 и number2 и
  результат присваиваем переменной result}
  result:= number1 + number2;

  {вывод примера на экран}
  Write (number1, '+', number2, '=', result);
End.            {признак конца программы}

```

Имя этой программы Sum_2. Оно отражает содержание программы, а также не содержит недопустимых символов.

Далее идет специально выделенный комментарий, в котором подробно указано условие задачи и автор программного кода решения.

Из разделов описаний имеется лишь один – раздел переменных. Он начинается со служебного слова Var. В нем описаны три переменные: number1, number2, result. Все они переменные целого типа. Поэтому они перечислены через запятую, после двоеточия указан тип значений переменных. Подобные объявления разделяются между собой точкой с запятой.

После описательной части идет раздел операторов, начинающийся со служебного слова Begin, после которого идут операторы языка.

Недостатком этой программы является ее неуниверсальность. То есть она вычисляет сумму конкретных чисел, значения переменных постоянны. А нам нужно научиться писать такие программы, которые решают поставленные задачи в общем виде, т. е. для любых значений переменных.

Для этого надо запрашивать значения у пользователя, анализировать их и выдавать соответствующий результат.

Задание.

1. Найдите в этой программе заголовок, раздел описания переменных, признак начала программы, признак конца программы, тело программы, комментарий.

2. Что обозначает строчка: `number1, number2, result: integer;`

3. Как вы понимаете запись: `number1 := 3;`

4. Чему равно значение переменной `result` после выполнения оператора: `result:= number1 + number2;`

5. Переведите с английского языка слово `Write`. Как вы думаете, что должен делать оператор с таким названием?

6. Поменяем местами второй и третий операторы. Будет ли программа работать? Почему?

7. Какой недостаток Вы видите у этой программы? Как нужно изменить условие задачи, чтобы решать подобные задачи с любыми числами. Подумайте, что надо изменить в теле программы, чтобы выполнить эту задачу.

1.5. Переменные, простые типы данных, оператор присваивания

Данные, которыми оперирует программа, могут быть определены в ней как *неизменные*, либо как *способные изменять свое значение* в ходе выполнения программы. Первые называются *константами*, а вторые *переменными*.

В программе *переменные* должны быть описаны в разделе `Var` (от слова `VARIABLE` – переменная), а *константы* – в разделе `Const`.

У всякой величины имеется три основных свойства: имя, значение, тип. Любая константа или переменная, использованная в программе, принадлежит к определенному типу. Тип задает множество допустимых значений переменных, возможные операции над значениями.

Тип Integer (целый)

Значения величины типа `Integer` лежат в диапазоне от `-2 147 483 648` до `+2 147 483 647`. Константы целого типа записываются в виде последовательности цифр со знаком или без него.

Переменные должны быть описаны в разделе `Var`:

```
Var    <имя_переменной> : Integer;
```

Над величинами целого типа допустимы *арифметические операции*: + (сложение), – (вычитание), * (умножение), Div (деление нацело), Mod (нахождение остатка от целочисленного деления).

Все операции дают результат *целого типа*. Например, $15 \text{ Div } 4 = 3$, $25 \text{ Mod } 4 = 1$.

Над целыми разрешено и *обычное деление*, оно обозначается косой чертой «/» и дает *результат вещественного типа*.

Тип Real (вещественный)

Константы вещественного типа (числа с дробной частью) изображаются *с десятичной точкой*: 12.3, –1.5, –0.75 или в *показательной форме*: –0.45E5, 6.7E–10, 0.355E6.

Для перезаписи числа из показательной в обычную форму надо *перенести запятую* на число разрядов, указанных после E, вправо, если число положительное, или влево, если отрицательное.

Например, $-0.45E5 = 45000$;
 $6.7E-10 = 0.00000000067$;
 $0.355E6 = 355000$.

Вещественные переменные требуют описания следующего вида:

```
Var    <имя_переменной> : Real;
```

Над величинами вещественного типа допустимы *арифметические операции*: + (сложение), – (вычитание), * (умножение), / (деление). Результат операций имеет *вещественный тип*.

Оператор присваивания

Оператор присваивания придает переменной конкретное значение, например: $X := 2$; $Y := 5$; одновременно *уничтожая старое*.

Формат команды: <имя переменной> := <выражение>;

Присваивать можно значение другой переменной или значение константы или результат вычисления выражения:

```

A:=B;
A:=B+C;
X:=Y+2-Z.

```

Задача 3. Поменять между собой значения двух переменных A и B , воспользовавшись третьей переменной R для временного хранения значения.

```

Program Change;
Var   a, b, r: Real;
Begin
  Write('Введите два числа ');
  Readln(a,b);
  R:=a;
  A:=b;
  B:=r;
  Write(' a=' , a, ' b=' , b);
End.

```

1.6. Консольный ввод-вывод

Оператор ввода

Ввод значений переменных с клавиатуры и вывод их на экран осуществляется следующими операторами:

<code>Read(список переменных);</code>	Ввод значений переменных через пробел или <Enter>; список переменных – последовательность имен переменных, разделенных запятыми
<code>ReadLn(список переменных);</code>	Ввод с последующим переводом курсора на новую строку
<code>Read;</code>	Приостановка выполнения программы до нажатия <Enter>
<code>ReadLn;</code>	Приостановка выполнения программы до нажатия <Enter> с последующим переводом курсора на новую строку

Пример. `Read(x)` – введенное число помещается в оперативную память, в отведенную ячейку, имеющую имя *x*.

Если список ввода содержит несколько имен, то для каждого надо ввести свое значение. Вводимые числа разделяют пробелами или нажатием клавиши <Enter>.

После работы этого оператора курсор располагается за последним введенным символом, но *не переводится* на новую строку.

Для перевода курсора на новую строку после ввода данных используется оператор `ReadLn(список переменных);` (Ln – сокращение Line – строка).

Оператор `ReadLn` отличается от `Read` еще и тем, что, введя необходимое количество данных, игнорирует «лишние». Другими словами, если ввести значений больше, чем переменных в операторе `Read`, то оставшиеся не принятые значения автоматически (!) будут переданы следующему оператору ввода (если таковой имеется дальше в программе). А вот `ReadLn` избыточные значения просто игнорирует, никуда не передавая.

Оператор вывода

Вывод значений переменных на экран осуществляется следующими операторами:

<code>Write(список вывода);</code>	Вывод
<code>WriteLn(список вывода);</code>	Вывод с последующим переводом курсора на новую строку
<code>WriteLn;</code>	Перевод курсора на новую строку

Список вывода может содержать имена переменных, числовые и текстовые константы, выражения. Элементы в списке разделяются запятыми. Если указана переменная, то на экран выводится ее значение. Константа выводится без изменения. Значения выражений вначале вычисляются, а затем выводятся на экран. Вывод происходит в том месте экрана, где находится курсор.

Если надо *перевести курсор на новую строку после вывода*, то применяется оператор `WriteLn(список вывода)`.

Вслед за выводимым выражением после двоеточия можно указать *ширину поля* экрана, в котором разместится выводимое значение.

Примеры.

```
Write(10:3, 55:6);
```

Если заданная ширина вывода окажется **меньше** выводимого значения, то ширина будет автоматически увеличена до нужного количества позиций.

10 55 (подчеркивание означает пробел, пустую позицию экрана)

```
WriteLn(10:0, 55:1);
```

1055

При выводе *вещественных значений* можно указать, сколько десятичных цифр следует выводить в дробной части числа. Количество цифр указывается вслед за шириной поля после двоеточия.

Например, если $X=3.14159$, а $Y=2.71468$, то оператор **WRITE(X:6:2,Y:8:3)** высветит на экране 3.14 2.715

Чтобы прокомментировать выводимые значения, в список вывода можно помещать *строковые константы* – текст, заключенный в апострофы (одинарные кавычки). Например, `Write('Ответ:', X:4, 'км/сек.');`

Так при $X=3.5$ этот оператор выведет: *Ответ: 3.5 км/сек.*

Задача 4. Написать программу поиска суммы, произведения и разности двух чисел.

Для каждого из чисел надо задать *имя переменной* и указать ее *тип*. Затем *ввести значения* этих переменных и, используя возможности оператора вывода, напечатать результаты.

```
Program Print;
Var  a,b,           {Вводимые числа - тип Вещественный}
     x,y,z: Real;   {Результаты - тип Вещественный }
Begin
  Write('введите два числа через пробел, затем нажмите
<Enter>');
  Readln(a,b);
  X:=a+b;
  Y:=a*b;
  Z:=a-b;
  Writeln('a+b=',x);
```

```
Writeln('a*b=',y);      {вывод      неформатированный      –  
                        в показательной форме}  
Writeln('a-b=',z:8:2);  {вывод      форматированный      –  
                        в десятичной форме}  
  
End.
```

Обратите внимание, в каком виде выводится значение *вещественной* переменной: *если указывается формат вывода* (то есть ширина поля и ширина дробной части), то вывод осуществляется в *десятичной форме*, *если формат не указывается – в показательной форме*.

2. Ветвление

2.1. Условный оператор

Алгоритм разветвляющейся структуры – это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия. В программировании ветвление реализуется с помощью условного оператора.

Условный оператор может использоваться в полной и сокращенной формах.

Формат полной формы:

If <условие> **Then** <оператор1> **Else** <оператор2>;

В этом случае при выполнении условия исполняется оператор1, при невыполнении – оператор2.

Замечания:

1) Перед зарезервированным словом Else точка с запятой не ставится, так как оператор еще не закончился.

2) Оператор1 и оператор2 могут представлять составные операторы, в этом случае они заключаются в операторные скобки Begin ... End.

Формат сокращенной формы:

If <условие> **Then** <оператор1>;

В этом случае при выполнении условия исполняется оператор1, в противном случае исполняется оператор, следующий за условным оператором.

Задача 5. Написать программу вычисления наибольшего из значений функций:

$$y_1 = x \cdot x + 1, \quad y_2 = 7 - x \cdot x, \quad y_3 = x + 1$$

для любого значения аргумента x .

```

Program task_5;      {программа разветвляющейся структуры}
Var  x , y1 , y2 , y3 , max : Real;
Begin
  Writeln ('Введите x');
  Readln (x);
  y1:=x*x+1;
  y2:=7-x*x;
  y3:=x+1;
  If  y1 > y2 Then  max:=y1 Else max:=y2;
  If  y3 > max Then  max:=y3;
  Writeln('y1=',y1:6:2,', y2=',y2:6:2,' y3=',y3:6:2,);
  Writeln('при x=',x:6:2,' наибольшее значение функции
=' ,max:6:2);
End.

```

2.2. Оператор вариантов

Если количество разветвлений программы больше двух, то используется оператор вариантов Case, который является более общим случаем условного оператора.

```

Case <выражение> Of
  P1: <оператор1>;
  P2: <оператор2>;
  . . .
  PN: <операторN>;
  Else <операторN+1>
End;

```

Оператор Case выполняет один из нескольких (как правило, больше двух) операторов в зависимости от значения выражения. Если значение выражения не совпадает ни с одним из значений $P1 \dots PN$, то выполняется оператор после Else или следующий после Case, если Else отсутствует.

Задача 6. Написать программу, определяющую время года по введенному с клавиатуры номеру месяца.

```

Program task_6;      { Множественный выбор }
Var  num: integer;

```



```
Begin
Writeln ('Введите номер месяца');
Readln(num);
Write('Время года: ', num);
Case num Of                { выбор значений селектора num}
1, 2, 12: Writeln ('Зима');
  3..5: Writeln  ('Весна');
  6..8: Writeln  ('Лето');
  9..11: Writeln ('Осень');
Else Writeln('нет месяца с таким номером!')
End;                        { завершение оператора case}
Readln;
End.
```

В приведенной программе при вводе номера месяца от 1 до 12 на экран выводится соответствующее время года, и выполнение программы заканчивается. Если же номер месяца превышает 12 или меньше 0, то выводится сообщение о неверном вводе месяца, для чего служит зарезервированное слово.

3. Организация повторения

3.1. Детерминированные циклы

Алгоритм циклической структуры – это алгоритм, в котором происходит многократное повторение одного и того же фрагмента программы. Программа циклической структуры содержит один или несколько циклов. Выделяют *детерминированные циклы* – циклы с заранее известным числом повторений. Изменяющаяся в цикле переменная называется параметром цикла (счетчиком).

Формат оператора цикла со счетчиком:

For <переменная> := <выражение1> **To** <выражение2> **Do** <оператор>;

Переменная должна быть порядкового типа. Порядковыми называются все простые типы, значения которых можно расположить в возрастающем порядке. Выражение1 и выражение2 должны быть того же типа, что и переменная.

Чтобы цикл выполнялся хотя бы раз выражение1 должно быть не больше выражения2.

Выполнение начинается с вычисления значений выражения1 и выражения2. Затем переменная получает значение выражения1 и делается проверка, не превышает ли значение переменной выражения2. Если не превышает, выполняется оператор тела цикла. После завершения оператора переменная получает следующее по порядку значение, и все повторяется, начиная с проверки. Когда значение переменной становится равным выражению2, оператор выполняется последний раз.

Если надо выполнять не один, а несколько операторов, то надо использовать составной оператор, то есть использовать операторные скобки.

Возможен вариант оператора цикла **For**, когда переменная принимает последовательно убывающие значения.

For <переменная> := <выражение1> **Downto** <выражение2> **Do**
 <оператор>;

В этом случае, чтобы цикл выполнялся хотя бы раз, выражение1 должно быть не меньше выражения2.

Пример.

```
For C:=1 To 9 Do
    WriteLn('C=',C);
```

На экран будет выведен столбик цифр от 1 до 9, после чего начнет выполняться оператор, следующий за *WriteLn*.

Задача 7. Вычислить значение факториала N!

```
Program factorial;
Var i, n : integer;
f: integer;
Begin
    Write('Введите значение N ');
    Readln(n);
    f:=1;
    For i:=1 To n Do f:=f*i;
    Write(n, '!=', f);
    Readln;
End.
```

Задача 8. Напечатать ряд из повторяющихся чисел 20 в виде:

20 20 20 20 20 20 20 20 20 20

```
Var i: byte;
Begin
    For I:=1 To 10 Do Write(20, ' ');
    Readln;
End.
```

Задача 9. Напечатать числа следующим образом:

10 10.4

11 11.4

...

25 25.4

```

Var i: byte;
Begin
  For i:=10 To 25 Do Write(i, ' ', i+0.4:0:1);
  {при сложении целого i и вещественного 0.4 получаем
   вещественный результат, значит надо выполнить его
   форматирование при выводе на экран}
  Readln;
End.

```

С помощью цикла типа `For` удобно находить суммы, произведения, искать максимальные и минимальные значения и т.п. членов последовательности. При нахождении суммы ее начальное значение 0, затем в цикле к этой переменной прибавляется соответствующий член заданной последовательности. Начальное значение произведения 1, затем в цикле оно умножается на общий член последовательности.

Задача 10. Написать программу вычисления n первых чисел Фибоначчи:

$$F_1 = 1; F_2 = 1; \dots; F_n = F_{n-1} + F_{n-2}.$$

Например, $F_3 = F_2 + F_1 = 1 + 1 = 2$; $F_4 = 2 + 1 = 3$ и т.д.

```

Var    x, y, z, i, n : integer;
Begin
  Writeln ('Введите n');   Read (n);
  x:=1; y:=0;
  For   i:=1 To n Do
    Begin
      z:=x ; x:=x+y ; y:=z;
      Write ('    ', x );
    End;
  Readln;
End.

```

3.2. Итеративные циклы с предусловием

Формат оператора цикла с предусловием:

While <логическое выражение> **Do** <оператор>;

Оператор (*тело цикла*) будет повторяться *пока истинно* логическое выражение. Перед каждым повторением оператора значение логического выражения вычисляется заново.

Если необходимо повторить несколько операторов, их следует объединить в составной оператор, т.е. заключить в операторные скобки:

```
While <логическое выражение> Do
  Begin
    <оператор1>;
    <оператор2>;
    ...
    <операторN>;
  End;
```

Цикл может не выполниться ни разу, если условие при входе в него оказалось ложным.

Пример.

```
C:=1;
While C<10 Do
Begin
  WriteLn('C=',C);
  C:=C+1;
End;
```

На экран будет выведен столбик цифр от 1 до 9, после чего начнет выполняться оператор, следующий за End;

Задача 11. Программа подсчета суммы S первых 1000 членов гармонического ряда $1+1/2+1/3+1/4+\dots+1/N$.

```
Var  S:Real;
      N:integer;
Begin
  S:=0;
  N:=0;
  While n<1000 Do
  Begin
    N:=N+1;
    S:=S+1/N
  End;
```

```

    Writeln(s);
End.

```

Задача 12. Написать программу подсчета суммы S членов гармонического ряда $1+1/2+1/3+1/4+\dots+1/N$ с точностью $\varepsilon = 10^{-3}$.

То есть здесь заранее не известно, сколько членов ряда надо суммировать. Вычисления продолжать до тех пор, пока значение очередного члена ряда больше и равно заданной точности.

```

Var    S, r, exp : Real;
I :integer;
Begin
exp:=0.001;           {точность вычислений, т.е.
                       минимальное значение слагаемого r,
                       которое надо учитывать при
                       суммировании}

S:=0;                 {сумма}
i:=1;                 {номер слагаемого}
r:=1/i;               {значение i-го слагаемого}
While r>=exp Do       {пока слагаемое не меньше заданной
                       точности}

Begin
    S:=S+r;
    i:=i+1;
    r:=1/i;
End;
Writeln('Сумма: ',s:0:3);
Writeln('Количество слагаемых: ',i-1);
Writeln('Значение последнего слагаемого: ',1/(i-1):0:6);
Writeln('Значение следующего слагаемого (не учитываемого):
',r:0:6);
Readln;
End.

```

Задача 13. Написать программу вычисления наибольшего общего делителя двух натуральных чисел A и B .

Воспользуемся для этого алгоритмом Евклида: будем уменьшать каждый раз большее из чисел на величину меньшего до тех пор, пока оба числа не станут равны.

```

Var    a,b : Integer;
Begin
  Write ('Введите два натуральных числа')
  Readln(a,b)
  While a<>b Do
    If a>b Then a:=a-b
      Else b:=b-a;
  Writeln('НОД=',a);
  Readln
End.

```

3.3. Итеративные циклы с постусловием

Повторение группы операторов (тела цикла) можно организовать и с помощью оператора, где проверка условия осуществляется после выполнения тела цикла.

```

Repeat
  <Оператор1>;
  <Оператор2>;
  ...
  <операторN>
Until <логическое условие>;

```

В отличие от оператора `While` в операторе `Repeat` сначала выполняется тело цикла, а затем проверяется условие. То есть в отличие от оператора `While` вычисление логического выражения происходит не до, а после очередного повторения цикла. Из-за этого цикл `Repeat` обязательно выполнится хотя бы раз, а цикл `While` может не выполниться ни разу.

Если условие в цикле `While` является условием продолжения повторений, то условие в цикле `Repeat` – условием выхода из цикла, его завершения. Поэтому для одной и той же задачи эти условия противоположны.

Тело цикла, заключенное между служебными словами `Repeat` и `Until`, **повторяется, если** логическое выражение, стоящее после слова `Until`, имеет значение **ложь**. Как только оно станет **истинно**, **цикл заканчивается**, и начнет выполняться оператор, стоящий после логического условия.

Между словами Repeat (повторить) и Until (до тех пор, пока) можно записать любое количество операторов без использования операторных скобок.

Пример. (Сравните с примером из раздела про цикл While).

<pre>C:=1; Repeat WriteLn('C=',C); C:=C+1; Until C=10;</pre>	<p>На экран будет выведен столбик цифр от 1 до 9, после чего начнет выполняться оператор, следующий за условием C=10.</p>
--	---

Задача 14. Составить программу подсчета суммы S первых 1000 членов гармонического ряда $1+1/2+1/3+1/4+\dots+1/N$, используя оператор цикла Repeat.

```
Var    S : Real;
        N : Integer;
Begin
  S:=0;
  N:=0;
  Repeat
    N:=n+1;
    S:=s+1/n
  Until n>1000;
  Writeln(s);
  Readln
End.
```

Задача 15. Найти наибольшее из N введенных с клавиатуры чисел.

Необходимо ввести с клавиатуры N чисел, найти из них наибольшее и вывести его. Для решения этой задачи предлагается следующий алгоритм:

1. Ввести первое число в переменную Max.
2. Ввести следующее число в переменную Next.
3. Если $Next > Max$, то $Max := Next$.

Пункты 2 и 3 повторять, пока не будут введены все числа.

4. Вывести значение переменной Max.


```

Var      N, max, next, k: integer;
Begin
  Write('Введите количество чисел'); Readln(n);
  Write('Введите число'); Readln(max);
  k:=1;
  repeat
    Write('Введите число');
    Readln(next);
    K:=k+1;
    If next>max Then max:=next
  Until k=n;
  Writeln(max);
  Readln
End.

```

Задача 16. Определить число n , при котором сумма квадратов натурального ряда чисел от 1 до n не превысит величину K , введенную с клавиатуры. Т.е. $S \geq K$, где $S = \sum_{i=1}^n i^2 = 1^2 + 2^2 + \dots + n^2$.

```

Var  k, s, n : Integer;
Begin
  Writeln( 'Введите K' );
  Readln (k);
  s:=0;    n:=1;
  Repeat
    s :=s+n*n;
    n := n+1;
  Until  s> k;
  Writeln('N= ', n : 3, '   s= ', s : 5 );
  Readln;
End.

```

Цикл повторяется до тех пор, пока условие, записанное после ключевого слова **Until**, будет ложным (не выполняется). Как только это условие выполнится, произойдет выход из цикла. После окончания цикла производится печать результата (оператор **Writeln**).

4. Основные операции работы с текстовыми файлами

Ввод информации можно осуществлять не только с клавиатуры, но и из файла. И выводить – можно не только на экран, но и в файл. При работе с файлами программа должна содержать следующее:

1. Описание файловой переменной в разделе описания программы (например, *F*):

```
Var F: Text;
```

2. Связывание файловой переменной с конкретным файлом – с помощью процедуры `Assign(F, 'имя_файла')`.

Например:

```
Assign(f, 'test.txt');           name := 'test.txt');
Assign(f, 'c:\user\test.txt');  Assign (f, name);
```

3. Открытие (создание) файла – с помощью одной из процедур:

- `Rewrite(F)` – для создания файла и для последующего вывода в него.
Если файл с указанным в `Assign` именем существует, то содержимое его теряется!
- `Append(F)` – открытие существующего файла для добавления записей.
- `Reset(F)` – открытие существующего файла для чтения записей.

При попытке открыть несуществующий файл, будет выдана ошибка.

4. Вывод в файл и ввод из файла.

Содержимое файла – это набор записей. Запись это, например, строка между Enter и Enter. Запись можно выводить (записывать) в файл и вводить (читать) из файла.

Для вывода в файл, – так же как и на экран, – используется процедура:

```
WriteLn(F, <список_выводимых_переменных_и_констант>);
```

Для ввода записей из файла, – так же как и с клавиатуры, – в строковую переменную используется процедура:

```
ReadLn(F, <строковая_переменная>);
```

Для последовательного чтения записей из файла, используется цикл, в котором для проверки, закончился ли файл, используется функция `EOF(F)`. Если файл закончился, то эта функция возвращает логическое значение **ИСТИНА**, иначе – **ЛОЖЬ**. И наоборот: выражение `Not EOF(F)` принимает значение **ИСТИНА**, если файл еще не закончился.

Пример:

```
While Not EOF(f) Do
Begin
    ReadLn(f, s);      {в строковую переменную S читается
                        очередная запись файла}
...
End;
```

5. Заккрытие файла – с помощью процедуры `Close(F)`.

Задача 17. Написать программу, которая создает файл *FILE.TXT* и записывает в него таблицу со значениями X и $Y=1/X$ на отрезке X от 1 до 10.

Var	f : Text;	Объявление файловой переменной <i>f</i>
	x : Byte;	
	y : Real;	
Begin		
	Assign(f, 'file.txt');	Связывание файловой переменной <i>f</i> с файлом <i>file.txt</i>
	Rewrite(f);	Создание пустого файла <i>file.txt</i>
	WriteLn(f, ' X Y=1/X');	Запись в файл заголовка таблицы
	For x := 1 To 10 Do	
	Begin	
	y := 1/x;	
	Writeln(f, x : 2, y : 6 : 2);	Запись в файл значений <i>x</i> и <i>y</i>
	End;	
	Close(f);	Закрытие файла
	End.	

Задача 18. Написать программу, которая добавляет в конец файла *FILE.TXT* таблицу со значениями X и $Y=\sin(X)$ на отрезке X от 1 до 10.

Var f : Text;	Объявление файловой переменной <i>f</i>
x : Byte;	
y : Real;	
Begin	
Assign(f, 'file.txt');	Связывание файловой переменной <i>f</i> с файлом <i>file.txt</i>
Append(f);	Открытие файла для добавления записей
WriteLn(f);	Запись пустой строки в файл
WriteLn(f, ' X Y=SIN(X) ');	Запись в файл заголовка таблицы
For x := 1 To 10 Do	
Begin	
y := sin(x);	
WriteLn(f, x :2, y :6 : 2);	Запись в файл значений <i>x</i> и <i>y</i>
End;	
Close(f);	Закрытие файла
End.	

Задача 19. Написать программу, которая читает записи из файла *FILE.TXT* и выводит их на экран.

Var f : Text;	Объявление файловой переменной <i>f</i>
s : String;	
Begin	
Assign(f, 'file.txt');	Связывание файловой переменной <i>f</i> с файлом <i>file.txt</i>
Reset(f);	Открытие файла для чтения
while Not EOF(f) Do	Пока не достигнут конец файла
Begin	
ReadLn(f, s);	Прочитать из файла строку в переменную <i>s</i>
WriteLn(s);	Вывод на экран строки
End;	
ReadLn;	
Close(f);	Закрытие файла
End.	

5. Задачи для самостоятельного решения

1. Составить программу для вычисления времени t встречи автомобилей, движущихся равноускоренно навстречу друг другу, если известны их скорости V_1 и V_2 , ускорения a_1 и a_2 и начальное расстояние S между ними. Расстояние S_1 пройденное первым автомобилем, вычисляется по формуле $S_1 = V_1 t + \frac{a_1 t^2}{2}$; расстояние S_2 , пройденное вторым автомобилем,

вычисляется по формуле $S_2 = V_2 t + \frac{a_2 t^2}{2}$. Время t встречи автомобилей

определяется из уравнения $V_1 t + \frac{a_1 t^2}{2} = S - (V_2 t + \frac{a_2 t^2}{2})$, откуда

$$t = \frac{-(V_1 + V_2) + \sqrt{(V_1 + V_2)^2 + (a_1 + a_2)2S}}{a_1 + a_2}.$$

2. Напечатать столбиком все целые числа от 20 до 35.

3. Напечатать столбиком квадраты всех целых чисел от 10 до b (значение d вводится с клавиатуры; $b \geq 10$);

4. Напечатать столбиком третьи степени всех целых чисел от a до 50 (значение a вводится с клавиатуры; $a \leq 50$);

5. Напечатать столбиком все целые числа от a до b (значения a и b вводятся с клавиатуры; $d \geq a$).

6. Распечатать в столбик таблицу умножения на 7.

7. Найти сумму S всех целых чисел, принадлежащих отрезку $[A, B]$. A и B предварительно ввести с клавиатуры.

8. Определить, сколько целых чисел принадлежит отрезку $[A, B]$. A и B предварительно ввести с клавиатуры.

9. Вводятся оценки по информатике каждого из студентов. В начале вводятся все пятерки, затем все остальные оценки. Сколько студентов имеют по информатике оценку «5»? Условный оператор не использовать.

10. Даны целые числа a и b ($a > b$). Определить:

а. Результат целочисленного деления a на b , не используя стандартную операцию целочисленного деления;

б. Остаток от деления a на b не используя стандартную операцию вычисления остатка.

11. Напечатать минимальное число, большее 200, которое нацело делится на 17.

12. Гражданин 1 марта открыл счет в банке, вложив 1000 руб. Через каждый месяц размер вклада увеличивается на 2% от имеющейся суммы. Определить: за какой месяц величина ежемесячного увеличения вклада превысит 30 руб.; через сколько месяцев размер вклада превысит 1200 руб.

13. В некоторой стране используются денежные купюры достоинством в 1, 2, 4, 8, 16, 32 и 64. дано натуральное число N . Как наименьшим количеством таких денежных купюр можно выплатить сумму N (указать количество каждой из используемых для выплаты купюр)? Предполагается, что имеется достаточно большое количество купюр всех достоинств.

14. Введите с клавиатуры 6 чисел и определите их среднее арифметическое.

15. Напишите программу, которая вводит целые числа с клавиатуры и складывает их, пока не будет введено число 0.

16. Напечатайте 20 первых степеней числа 2.

17. Найдите максимальное из N введенных с клавиатуры чисел.

18. Дано натуральное число. Выяснить, является ли оно простым, т.е. делится только на 1 и на само себя.

19. Вывести на экран содержимое файла с программой.

20. Содержимое файла *MAS1.PAS* вывести в файл *MAS11.PAS*. (То есть создать копию файла).

21. Вводимые с клавиатуры строки выводить в файл *FILE.TXT* – до тех пор, пока с клавиатуры не будет введена пустая строка.

22. В файле *FILE.TXT*, созданном в результате выполнения предыдущего задания, подсчитать: 1) количество строк, 2) сколько раз встречается буква 'А'.

23. Вводимые с клавиатуры *Фамилию*, *Имя* и *Отчество* выводить одной строкой в файл *NAMES.TXT* – до тех пор, пока с клавиатуры не будет введена «пустая» *Фамилия*.

24. В файле *NAMES.TXT*, созданном в результате выполнения предыдущего задания, подсчитать: 1) количество человек, 2) сколько раз встречается заданная фамилия.

Литература

1. Баженова И.Ю. Введение в программирование. Учебное пособие – М: БИНОМ. Лаборатория знаний, 2007.
2. Костюк Ю.Л., Фукс И.Л. Основы разработки алгоритмов. Элективный курс : учебное пособие – М: БИНОМ. Лаборатория знаний, 2009.
3. Окулов С.М. Основы программирования. – М.: ЮНИМЕДИСТАЙЛ, 2002.
4. Павловская Т.А. Паскаль. Программирование на языке высокого уровня. Учебник. – М: Просвещение, 2007.
5. Столяр С.Е., Владыкин А.А. Информатика: представление данных и алгоритмы – М: БИНОМ. Лаборатория знаний, 2007.
6. Сулейманов Р.Р. Методика решения учебных задач средствами программирования : методическое пособие – М: БИНОМ, 2010.
7. Шауцукова Л.З. ИНФОРМАТИКА. Практика алгоритмизации и программирования. Информатика 10–11. – М.: Просвещение, 2000.